GUI Script

Application Note ANGS1.00.01 [Issued date Jan 11, 2011]

GUI Script Example Project:
A Questionnaire on 4.3” LCD

1. Introduction

To illustrate how to build a project with GUI script, we had built a mini project that has all basics for writing GUI script and application software development.

The project theme is a simple questionnaire on 4.3” touch screen LCD. As same as the real implementation, first we have to design what would come out. Here all the screens that we will create are shown in the next chapter. This is just like when GUI is designed on a PC. Then we will create scripts along the screens in chapter 3. As the scripts cannot complete our project, we still need some code called application software and will be described in chapter 4. After reading this document, you will see how easy GUI script can be implemented and how they dramatically reduce source code needed.

2. GUI on Screens

The first screen (screen 0) is welcome screen. It uses a part of background image as a button.

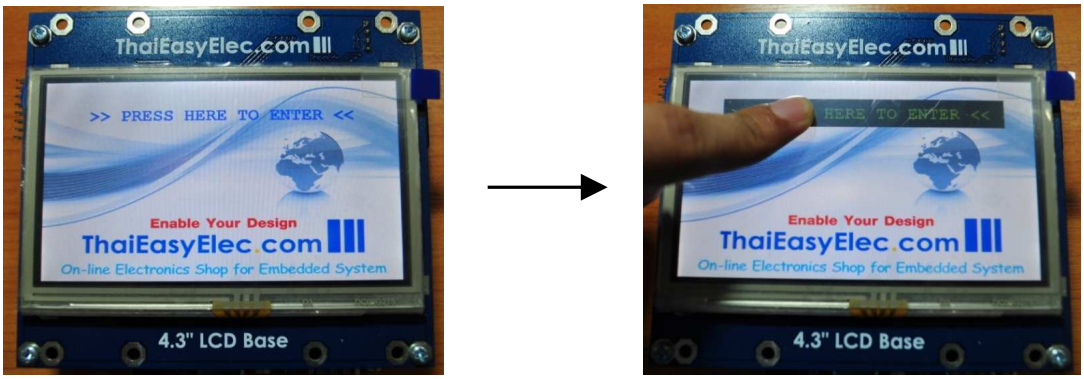

Figure 1: The first screen and when its button is touched

The second screen (screen 1) is a form that has textbox and on-screen keyboard. A question ‘Which country do you live?’ is shown as a label. Here we enter a name of country and the application software has to collect it.

Figure 2: Screen 1

The third screen (screen 2) is a question with choices. Again, all texts here are labels. There are 2 checkboxes and users can check them both. As we don’t have checkboxes implement with GUI Script, we will use interlock-shift buttons. Then the answers will be collected by application software.

Figure 3: Screen 2

Figure 4: Pressing a checkbox

Figure 5: When choose 1 answer

Figure 6: When choose both answers

Figure 7: Press ‘Next’ button to go to screen 3.

The forth screen (screen 3) is a question with 3 checkboxes that can be all checked.

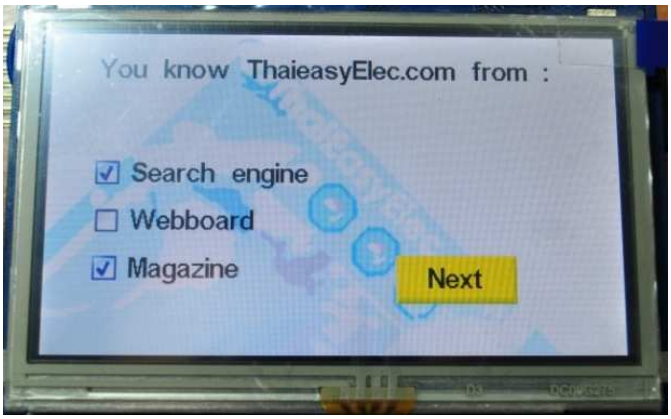

Figure 8: Screen 3

The fifth screen (screen 4) is a question and 3 choices with their images. Buttons now are simple. User can choose only one answer to vote and a textbox will show the answer you choose. Before choosing an answer, the submit vote button is disabled. Choosing any answer will enable the vote button. Now the application software will store the answer by reading the textbox.

Figure 9: Screen 4

Figure 10: First pressing (Please note that 'VOTE' button is in disable state)

Figure 11: The button is released

Figure 12: Pressing other choice

Figure 13: Second button released

Figure 14: Third pressing

Figure 15: Third button released

Figure 16: Press ‘VOTE’ button to submit answer

The last screen (screen 5) shows a table with vote result. Here the application software reads value from variables and write to the result table. 5 seconds after the screen showed, it turns back to the first screen (with accumulated result until the board is reset).

Figure 17: Screen 5

3. Scripts

Please read GUI Script user manual along with this chapter, there are descriptions in detail about what each script command used for.

First, you have to register images, buttons, fonts and sound (but we have no sound in this example). Registering commands are shown in figure 18.

```
########### BG ############
GUI RBG  0 2 1 bg/BG00.bmp
GUI RBG  2 2 1 bg/BG2.bmp
GUI RBG  3 2 1 bg/BG3.bmp
GUI RBG  4 2 1 bg/BG4.bmp
GUI RBG  5 2 1 bg/BG5.bmp

########### Button ##########
GUI RIMG  0 2 1 buttons/uncheck.bmp
GUI RIMG  1 2 1 buttons/checked.bmp
GUI RIMG  2 3 390 40 255 0x07FF
GUI RIMG  3 3 100 40 255 0x07FF
GUI RIMG  4 2 1 buttons/SUN7_BT.bmp
GUI RIMG  5 2 1 buttons/SUN7_BT_p.bmp
GUI RIMG  6 2 1 buttons/Bluebee_BT.bmp
GUI RIMG  7 2 1 buttons/Bluebee_BT_p.bmp
GUI RIMG  8 2 1 buttons/Arduino_BT.bmp
GUI RIMG  9 2 1 buttons/Arduino_BT_p.bmp
GUI RIMG  10 2 1 buttons/vote_d.bmp
GUI RIMG  11 2 1 buttons/vote_n.bmp
GUI RIMG  12 2 1 buttons/vote_p.bmp
GUI RIMG  13 2 1 buttons/cancel_d.bmp
GUI RIMG  14 2 1 buttons/cancel_n.bmp
GUI RIMG  15 2 1 buttons/cancel_p.bmp

########### Image ##############
GUI RIMG  16 2 1 img/SUN7.bmp
GUI RIMG  17 2 1 img/Bluebee.bmp
GUI RIMG  18 2 1 img/Arduino.bmp

################FONT#######################
GUI RFONT 0 25 24 1 fonts/font_arial_eng25x24.bin
GUI RFONT 1 30 32 2 fonts/font_arial_eng30x32.bin
GUI RFONT 2 16 16 1 fonts/font_arial_eng16x16.bin
GUI RFONT 3 24 24 1 fonts/font_arial_eng24x24.bin
```

Figure 18: Registering images, buttons and fonts.

The first screen has only two components. Background image is shown in figure 19.

Figure 19: BG00.bmp, registered as background image 0.

Please note that there is an on-button message already in the background. You can make a simple effect by inverting color in the button area as shown in figure 20.

```
GUI SCS S 0
GUI BG  A 0

GUI BT  O 50 27
GUI BT  N 2
GUI BT  t
GUI BT  i
GUI BT  G 1
GUI BT  E
```

Figure 20: Making button area with invert-color effect

To begin each screen, you must assign screen ID at the beginning of them by using `'GUI SCS S xxx'` command. If you have background image, use `'GUI BG  A xxx'` to add an image as background.

A plain image is created first with `'RIMG 2 3390 40 … '`. Then it's used as normal image for our button (390x40 is its size). The button is placed at the location where it covers the area '>> PRESS HERE TO ENTER <<' on the background. Then, I had set transparent and inverse mode (`'BT t'` and `'BT i'`). So you will see only background image if it is not touched and the area will be inversed when you touch it.

Figure 21: Inversed color in button area

Pressing the button leads us to screen 1. And that is the result of ‘GUI BT G 1’.

The second screen shows how to create a keyboard.

```
GUI SCS S 1
GUI SCS K 2
GUI BG  C 0xFFFF

GUI TXT O 40 70
GUI TXT S 250 40
GUI TXT F 0
GUI TXT L 10
GUI TXT I
GUI TXT C 0 0xFFFA
GUI TXT A 0
GUI TXT K
GUI TXT E

GUI BT  O 290 70
GUI BT  N 3
GUI BT  T Submit
GUI BT  M 2 2
GUI BT  G 2
GUI BT  E

GUI LBL O 40 10
GUI LBL A 0
GUI LBL F 0
GUI LBL T Which country do you live?
GUI LBL E
```

Figure 22: Screen 1 (second screen) script

Figure 23: On-screen-keyboard

As seen in figure 22 at the second line, I use only one command to create on-screen keyboard (`GUI SCS K 2`). The GUI engine library provides ready-to-use keyboard layout so there is no need to design keyboard yourself (but if you would like make it your style, there are a simple way and we will show it on the next paper).

The second paragraph of scripts in figure 22 creates a textbox that display characters you type. The command `'TXT K'` is the key script that links this textbox to keyboard.

The third paragraph creates a 'Submit' button which is a plain color button. We make it move 2 pixels downward and rightward when user presses it by using `'GUI BT M 2 2'`.

The forth paragraph creates a text label that have text message 'Which country do you live?' `'GUI LBL A 0'` causes the message to be left justified. `'GUI LBL F 0'` makes it displayed with Arial 25x24-pixel English font.

The third screen shows how to create checkboxes.

```
GUI BT  O 40 111
GUI BT  N 0
GUI BT  M 2 2
GUI BT  I 0
GUI BT  E
GUI BT  O 40 111
GUI BT  N 1
GUI BT  M 2 2
GUI BT  I 1
GUI BT  d
GUI BT  E
```

Figure 24: The first checkbox script

Figure 25: The first checkbox after a single touch

As seen in figure 24, I use interlock button to circulate 2 image buttons. One is an empty box and another is a mark-checked box. So it will display check mark when you press and the check mark will be removed when you press again. Note the `'BT d'`, this allows the image with N=0 to be shown first. Every time you create an interlock button, make sure to have only one part of it enabled.

The forth screen uses the same technique as the third screen.

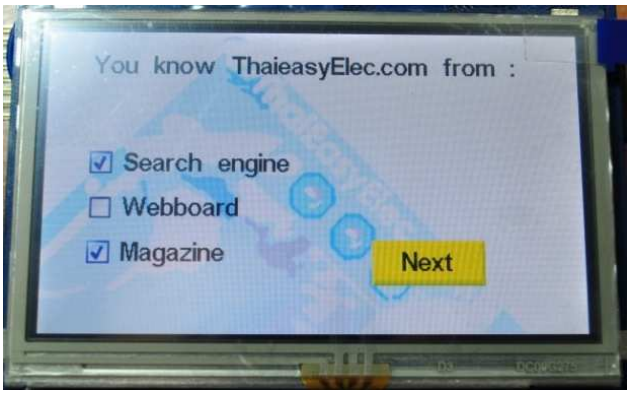

Figure 26: The forth screen

The fifth screen shows another type of button.

Figure 27: 3 buttons with ‘normal’ and ‘pressed’ status

```
GUI BT  M 2 2
GUI BT  S
GUI BT  O 7 160
GUI BT  N 4
GUI BT  P 5
GUI BT  n SUN7
GUI BT  E

GUI BT  L
GUI BT  O 165 160
GUI BT  N 6
GUI BT  P 7
GUI BT  n Bluebee
GUI BT  E

GUI BT  L
GUI BT  O 323 160
GUI BT  N 8
GUI BT  P 9
GUI BT  n Arduino
GUI BT  E
```

Figure 28: Scripts of buttons in figure 27

Scripts in figure 28 use 2 different images for ‘normal’ and ‘pressed’ states. It uses `‘GUI BT  N x’` and `‘GUI BT  P y’` to assign images for normal and pressed status. `‘GUI BT n xxx’` uses for naming the buttons. Application software can use this name to show in textbox or use it for other purpose.

This screen also shows how to use ‘save and load’ technique. In this example, `‘GUI BT S’` save only one command, that is `‘GUI BT M 2 2’`, but you can save more as in figure 29, 3 commands saved!

```
GUI BT  N 0
GUI BT  P 0
GUI BT  t
GUI BT  S
GUI BT  O 49 126
GUI BT  G 5
GUI BT  E
```

Figure 29: ‘Save’ command example

So you can reduce size of the script and edit button setting easily if you have many buttons with shared features.

The last button illustrates how to disable and enable pressing. It is ‘VOTE’ button.

```
GUI BT  O 350 195
GUI BT  D 10
GUI BT  N 11
GUI BT  P 12
GUI BT  M 1 1
GUI BT  t
GUI BT  d
GUI BT  G 5
GUI BT  E
```

Figure 30: ‘VOTE’ button script

Figure 31: Disable and enable button

As seen in figure 30, you have to use 2 commands to create button with disable status. The first is `‘GUI BT D x’`, used for assigning an image to be shown when disabled. Then, you have to disable the button by command `‘GUI BT d’` and you

can enable it by software. Note that after enabling by software, it won't be disabled again when the screen shown later.

After pressing 'VOTE' button, application software will record score in 3 variables. Each variable stores scores for each choice.

The last screen is the vote result. It displays a table that shows each product's score. The script in figure 32 shows how to create a table on SUN7's LCD screen.

```
GUI TAB O 20 70
GUI TAB c 0 250 0x07E0 0xffff Product
GUI TAB c 1 100 0x07E0 0xffff Score
GUI TAB R 40 4
GUI TAB A 1
GUI TAB E
```

Figure 32: Table script

Figure 33: Table on LCD screen

Please note that you can set text at the top row but you have to use software function to write the rest.

At the beginning of screen 5 (the sixth screen), you will see scripts as figure 34

```
GUI SCS S 5
GUI SCS W 50 0
GUI BG  A 5
```

Figure 34: Scripts at the beginning of screen 5

`'GUI SCS W xxx yyy'` is 'wait' command. The counter is increased every 100 milliseconds until it reaches 50 then go to screen 0. That is, you will see the result table for 5 seconds then the screen return to the start and a new round begins.

4. Application Software

Now, SUN7 has a script file and can run graphic function. But it cannot collect or process any data because it does not have application software to work with GUI script. This topic will guide you how to develop software to work together with it.

First, I have to define maximum screen object in header file (app_scr_func.h) as figure 35. These numbers depend on amount of object in script. And the rest lines are function declarations for 'app_scr_func.c'. Please note that this defining is just to make your objects have enough 'slots'. Using more (or equal) objects than their defined slots will be warned with red lines on the screen when `'GUI END'` found.

```
#define MAX_SCR_OBJ      40
//#define MAX_LANGUAGE   2
//#define MAX_PATH       8
#define MAX_SCR          140          //all screens
#define MAX_BACKGROUND   10           //all background
#define MAX_IMAGE        400          //all images
#define MAX_BUTTON       400          //all buttons
#define MAX_IMAGE_BOX    400          //all image boxed
#define MAX_TEXTBOX      50           //all textboxes
#define MAX_TABLE        1            //all tables
//#define MAX_TABLE_COL  1            //table column
#define MAX_TEXT_SIZE    4096         //text pool size
#define MAX_LABEL        50           //all label
#define MAX_BOX          1            //all boxes
#define MAX_FONT         5            //all fonts
#define MAX_EXT_FONT     1            //all ext fonts
#define MAX_IMG_FONT     10           //all image fonts

#define MAX_SOUND        30
#define MAX_SOUND_SEQ    20
#define MAX_SOUND_FILE   60

...
```

Figure 35: Values defined in 'app_scr_func.h'

Then, I create a new C file (app_scr_func.c) and include its header 'app_config.h' at the beginning of file and create global variables as shown in figure 36.

```
001 #include "../app_config.h"
002
003 char country[10];
004 int all_vote[5];
005 int sun7=0,bluebee=0,arduino=0;
```

Figure 36: File inclusion and global variables declaration

Figure 37 is the most important function, `AppScrInit`, it checks status and initialize all screens' data.

```
159  void AppScrInit(void)
160  {
161      unsigned char i;
162      memset (all_vote,0,5);
163
164      guisc[1].init = scr1_init;
165      guisc[1].kp_char_event = get_kb_scr_country;
166      guibt[GUIGetBtID(1,0)].release = save_country;
167
168      guisc[2].init = scr2_init;
169      //screen 2: 4 buttons are used
170      for(i=0;i<4;i++)
171          guibt[GUIGetBtID(2,i)].release = do_check_box;
172
173      guisc[3].init = scr3_init;
174      //screen 3: 6 buttons are used
175      for(i=0;i<6;i++)
176          guibt[GUIGetBtID(3,i)].release = do_check_box;
177
178      guisc[4].init = scr4_init;
179      guibt[GUIGetBtID(4,0)].release = update_txt_box;
180      guibt[GUIGetBtID(4,1)].release = update_txt_box;
181      guibt[GUIGetBtID(4,2)].release = update_txt_box;
182      guibt[GUIGetBtID(4,3)].release = cnt_vote;
183
184      guisc[5].init2 = show_result;
185  }
```

Figure 37: `AppScrInit` function

At line 162, '`memset (all_vote,0,5)`' clears value in variable array '`all_vote`' which has 5 members.

The code at line 164, `guisc[1].init = scr1_init;`, initialize screen 1 by calling `scr1_init` function to clear value in textbox of screen 1.

```
007  void scr1_init(void)
008  {
009      memset(guitxt[GUIGetTxtID(1,0)].text,0,10) ;
010  }
```

Figure 35: `scr1_init` function

Please note that it calls `GUIGetTxtID(1,0)` function to get textbox ID of **textbox 0** in **screen 1**.

At line 165, it calls `get_kb_scr_country` function.

```
036  void get_kb_scr_country (char c)
037  {
038      saych(c);
039      if(c==0x0D)
040      {
041          if(guitxt[GUIGetTxtID(1,0)].text[0])
042          {
043              strcpy(country,guitxt[GUIGetTxtID(1,0)].text);
044              GUIGotoScreen(2);
045          }
046      }
047  }
```

Figure 36: `get_kb_scr_country` function

The function named '`get_kb_scr_country`' will send character got from keyboard to serial port. It will copy text in textbox to the variable named '`country`' then go to screen 2 when user press ENTER button (ASCII code is 0x0D) and textbox is not empty.

However, user can press 'Submit' button which is **button 0** of **screen 1** instead of pressing ENTER. At line 166, `guibt[GUIGetBtID(1,0)].release = save_country;`, it calls `save_country` function when user releases button.

```
049  void save_country (void)
050  {
051      if(guitxt[GUIGetTxtID(1,0)].text[0])
052      {
053          strcpy(country,guitxt[0].text);
054          GUIGotoScreen(2);
055      }
056  }
```

Figure 37: `save_country` function

The function in figure 37 will copy text in textbox to the variable named 'country' and go to screen 2 if textbox is not empty.

At line 168, `guisc[2].init = scr2_init;`, screen 2 will be initialized by calling function named '`scr2_init`' as shown in figure 38.

```
018  void scr2_init(void)
019  {
020      GUIEnableBt(GUIGetBtID(2,0));
021      GUIDisableBt(GUIGetBtID(2,1));
022      GUIEnableBt(GUIGetBtID(2,2));
023      GUIDisableBt(GUIGetBtID(2,3));
024  }
```

Figure 38: `scr2_init` function

According to chapter 3 page 11, to prevent initial state of checkboxes problem, the function in figure 38 will enable button 0 and 2 of screen 2 that are the beginning state of checkbox (unchecked) and disable button 1 and 3 to prevents status that should not be at the beginning of screen.

```
170     for(i=0;i<4;i++)
171         guibt[GUIGetBtID(2,i)].release = do_check_box;
```

Figure 39: Storing score from checkbox.

Line 170-171 in figure 39 bind function that collects scores for choices in screen 2.

```
058 void do_check_box (void)
059 {
060     unsigned char num_bt;
061     num_bt = ScrGetCurrentObj();
062     if(scr_id == 2)
063     {
064         switch (num_bt)
065         {
066             case 1:
067                 all_vote[0]++;
068                 break;
069
070             case 3:
071                 all_vote[1]++;
072                 break;
073         }
074     }
075     else
076     {
077         switch (num_bt)
078         {
079             case 1:
080                 all_vote[2]++;
081                 break;
082             case 3:
083                 all_vote[3]++;
084                 break;
085             case 5:
086                 all_vote[4]++;
087                 break;
088         }
089     }
090 }
```

Figure 40: `do_check_box` function

`do_check_box` function in figure 40 uses `ScrGetCurrentObj` function to get number of button which is just released then increase value of variable array named '`all_vote[]`'. If the button is button 1 or 3 of screen 2 which is the button with check mark, value in `all_vote[0]` or `all_vote[1]` that represent 2 choices of screen 2 will be increased.

```
173     guisc[3].init = scr3_init;
174     //screen 3: 6 buttons are used
175     for(i=0;i<6;i++)
176         guibt[GUIGetBtID(3,i)].release = do_check_box;
```

Figure 41: Storing score from checkbox in screen 3.

Figure 41 is code that collects checkbox values in screen 3. Because screen 3 has the same style as screen 2, so the initial function '`scr3_init`' has the same style as '`scr2_init`' and when user release checkbox buttons, `do_check_box` function will be called to verify button status and increase value of `all_vote[2]` or `all_vote[3]` or `all_vote[4]` if current object is one of button 1,3 or 5 of screen 3.

```
026 void scr3_init(void)
027 {
028     GUIEnableBt(GUIGetBtID(3,0));
029     GUIDisableBt(GUIGetBtID(3,1));
030     GUIEnableBt(GUIGetBtID(3,2));
031     GUIDisableBt(GUIGetBtID(3,3));
032     GUIEnableBt(GUIGetBtID(3,4));
033     GUIDisableBt(GUIGetBtID(3,5));
034 }
```

Figure 42: `scr3_init` function

At line 178 of figure 37, `guisc[4].init` calls function named '`scr4_init`' to initialize screen 4.

```
012 void scr4_init(void)
013 {
014     memset(guitxt[GUIGetTxtID(4,0)].text,0,7) ;
015     GUIDisableBt(GUIGetBtID(4,3));
016 }
```

Figure 43: `scr4_init` function

`scr4_init` function in figure 43 clears value in textbox and disable **button 3** of **screen 4** which is 'VOTE' button at the beginning of screen 4 to prevent initial status problem.

```
179     guibt[GUIGetBtID(4,0)].release = update_txt_box;
180     guibt[GUIGetBtID(4,1)].release = update_txt_box;
181     guibt[GUIGetBtID(4,2)].release = update_txt_box;
```

Figure 44: The code at line 179-181

Figure 45: Button 0, 1, and 2 of screen 4.

As seen in figure 44, when user presses and releases button 0, 1 or 2, it calls function named '`update_txt_box`'.

```
092 void update_txt_box (void)
093 {
094     unsigned char num_bt;
095
096     num_bt = ScrGetCurrentObj();
097     switch (num_bt)
098     {
099         case 0:
100             strcpy(guitxt[GUIGetTxtID(4,0)].text,guibt[GUIGetBtID(4,0)].name);
101
102             GUIRefreshTxt(0);
103
104             break;
105
106         case 1:
107             strcpy(guitxt[GUIGetTxtID(4,0)].text,guibt[GUIGetBtID(4,1)].name);
108
109             GUIRefreshTxt(0);
110
111             break;
112
113         case 2:
114            strcpy(guitxt[GUIGetTxtID(4,0)].text,guibt[GUIGetBtID(4,2)].name);
115
116            GUIRefreshTxt(0);
117
118            break;
119
120     }
121
122     GUIEnableBt(GUIGetBtID(4,3));
123     GUIRefreshBt(3);
124 }
```

Figure 46: `update_txt_box` function

The `update_txt_box` function will display name of product which chosen in textbox by copying button's name to textbox and refresh it.

Because 'VOTE' button is disable at the beginning of screen 4 and enable when user choose any product, the code at line 122 enables the button.

```
182     guibt[GUIGetBtID(4,3)].release = cnt_vote;
```

Figure 47: Code at line 182.

As seen in figure 47, when user releases button 3 of screen 4 which is 'VOTE' button, function named '`cnt_vote`' will be called.

```
142 void cnt_vote (void)
143 {
144     if(strcmp(guitxt[GUIGetTxtID(4,0)].text,"SUN7")==0)
145     {
146         sun7++;
147     }
148     if(strcmp(guitxt[GUIGetTxtID(4,0)].text,"Bluebee")==0)
149     {
150         bluebee++;
151     }
152     if(strcmp(guitxt[GUIGetTxtID(4,0)].text,"Arduino")==0)
153     {
154         arduino++;
155     }
156     GUIGotoScreen(5);
157 }
```

Figure 48: `cnt_vote` function

In `cnt_vote` function, it uses 3 global variables that store each product's score. `strcmp` (string compare) function compares product name and string in textbox. Then, the related variable's value will be increased if string in the textbox match product name.

```
184     guisc[5].init2 = show_result;
```

Figure 49: Code at line 184.

At line 184, `guisc[5].init2` calls `show_result` function to draw a table and display product's score in screen 5.

```
126  void show_result (void)
127  {
128      char sun[10],blue[10],ardui[10];
129      GUIWriteTable(0,1,0,guibt[GUIGetBtID(4,0)].name);
130      GUIWriteTable(0,2,0,guibt[GUIGetBtID(4,1)].name);
131      GUIWriteTable(0,3,0,guibt[GUIGetBtID(4,2)].name);
132
133      int_to_str(sun7,(unsigned char *)sun);
134      int_to_str(bluebee,(unsigned char *)blue);
135      int_to_str(arduino,(unsigned char *)ardui);
136
137      GUIWriteTable(0,1,1,sun);
138      GUIWriteTable(0,2,1,blue);
139      GUIWriteTable(0,3,1,ardui);
140  }
```

Figure 50: `show_result` function

The `show_result` function writs products' name and their scores to the table in screen 5.

As seen in figure 50 at the line 128, I create 3 character string variables to store product's score as character type. This function uses `int_to_str` function to convert integer value in variable to character-type number.

Then `GUIWriteTable` function writs text message to the specified table cell.

If user answers questions again, the result in table will change because product's score are stored as global variables.

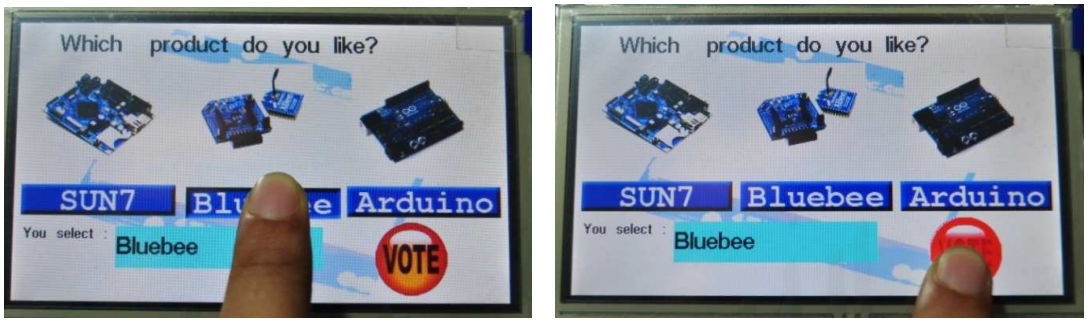

Figure 51: Answer other choice

Figure 52: Data in table change because of value in variable change

Prepared by
ThaiEasyElec.com
Venus Supply Co.,Ltd
196/1, Soi Thedsaban-Nimit-Nau 8,
Thedsaban-Nimit-Nau Road, Ladyao,
Chatuchak, Bangkok 10900
Tel. +(66)2954-2408 , Fax. +(66)2953-8443
Email Sale@thaieasyelec.com
Support@thaieasyelec.com
Info@thaieasyelec.com